

METHOD FOR ELIMINATING OR REDUCING HANG CONDITIONS IN COMPUTER SYSTEMS

5

BACKGROUND OF THE INVENTION

Technical Field

This invention relates to a method for eliminating or reducing hangs during a shutdown or a panic of computer systems and improves the validity of debugging data acquired from the panic shutdown.

10

Description of the Prior Art

Arguably the most important part of a computer system is the data stored on it. This data is almost exclusively stored on offline storage devices such as disks. It seems to be a continuing trend that the speed of computers remains ahead of the speed of offline storage, and for these reasons most implementations of computer systems use a caching mechanism to keep frequently/recently accessed data in memory rather than on disk. There is a risk with this caching, namely that if anything prevents the cached data from being written to offline storage, the data will be lost. As an example, a sudden loss of power can result in cached data being lost. Another source of cache loss is a failure of the operating system (OS) that results in sudden shutdown without flushing (writing to offline storage) of the caches.

Symmetric Multi-Processor (SMP) computer systems use mutual exclusion (mutex) mechanisms to coordinate (prevent collision) of the activities being performed by the various processors (threads of execution). The integrity of these mutexes must be maintained, or severe problems will result including data corruption. Additionally, to prevent problems such as deadlock (system hangs) or system slowdown, it must be ensured that threads which acquire mutexes, do so for a minimal time and according to well-defined rules set forth by the designers of the code. During normal system operation, once the code design has been completed and the implementation sufficiently debugged, these mutexes tend to operate quite well. However, there are some cases where the operation of these mutexes can be disrupted, most notably during system shutdown but also as a result of "bugs" (in both hardware and software), and in general instances of system failure. Typically, the most reliable mechanism for recovering from these disruptions, is to shutdown the system and restart it from scratch.

Panic is a special case of OS shutdown. During a normal, orderly shutdown, the OS has the luxury of gracefully terminating applications and subsystems, including database systems and file systems, which results in complete data preservation (i.e. zero data loss or corruption). By

definition, a panic is the expedited shutdown of the OS and system after detecting a failure of hardware or software. This creates a hostile environment in which the panic shutdown must operate. There are two main goals of a panic:

- 5 1. Minimize corruption and data loss, mainly by synchronization of in-memory (cached) data with off-line storage (disks).
2. Accurately preserve the state of the software and hardware in order to facilitate diagnosis of the root cause of the problem.

10 Note that these two goals may be in conflict under certain circumstances, for example, if a panic is the result of a failure in the disk subsystem. In fact, some circumstances may make it impossible to accomplish either, or both, of these goals.

Another intent of a panic is to restart the system with the hope that a fresh start will allow the system to run more successfully. To this end, the time taken by the panic shutdown, core dump, and reboot is critical to the availability of the system and must be minimized. With the unpredictable nature of the system during a panic, there are many impediments to completing the shutdown that may be encountered. One significant impediment is due to the fact that applications have not terminated (gracefully) and thus have not released the mutexes that they may have been holding. The panic shutdown must resolve these mutexes without adding to data corruption, more specifically it must ensure that "dead" mutexes are broken promptly while mutexes held by legitimately running threads are not "stolen".

25 Aahlad, U.S. Patent Number 5,907,675, provides a method for managing deactivation and shutdown of a server. The patent teaches a method for a server to gently exit without actually terminating the server's clients. There is a lock-up flag that is used to prevent any new clients from joining. The invention implements an orderly and predictable server deactivation and/or shutdown. The invention utilizes an "usher" that continuously maintains a transaction counter indicative of the number of clients that are actively utilizing services.

30 Hapner, U.S. Patent Number 5,940,827, provides a method for using mutexes to protect the shutdown time commit phase of a database and the shared database cache. The system informs the shutdown thread that the time commit thread is going away. The mutex is created to correspond to a piece of code.

35 However, while Aahlad and Hapner disclose shutting down an application, the present invention relates to shutting down the operating system that supports these applications. The difference between the prior art and the present invention is analogous to turning off one's television set verses shutting off the power for the whole house.

Therefore, there remains an unmet need for expediting panic shutdowns by eliminating or reducing hangs while minimizing data corruption/loss and processing system state information.

SUMMARY OF THE INVENTION

One aspect of this invention is an improved mechanism for dealing with mutual exclusion (mutex) structures in a computer system which can have at least one possible state transition. The mechanism includes an identifier, an indicator, and a mutex handler. The identifier identifies the owner of the mutex, which is preferably a processor or process. The indicator shows whether the mutex was acquired before or after the state transition. Where the state transition is from normal operation of the computer system to a "panic" shutdown, the indicator shows whether the mutex was acquired pre- or post- panic. The mutex handler is responsive to the identifier and the indicator, and in the preferred embodiment includes first routines for use before the state transition, and second routines for use after the state transition.

Another aspect of the invention is a method for handling a mutex after a state transition in the computer system. The method determines whether the mutex was acquired before or after the state transition, and handles the mutex differently depending upon whether the mutex was acquired before or after the state transition. In a preferred embodiment, the mutex determines from a data structure whether the mutex was acquired before or after the state transition.

Yet another aspect of the invention is an article of manufacture comprising a computer-readable signal bearing medium such as a recordable data storage medium or a modulated carrier signal. Means in the medium determine whether the mutex was acquired before or after a straight transition in the computer system, and handle the mutex differently depending upon whether the mutex was acquired before or after the state transition.

The invention thus adapts to panic shutdowns and other state transitions in a computer system, distinguishing between pre- and post-transition mutexes, and handling them differently so as to minimize corruption and data loss and accurately preserve a system state. Other features and advantages of this invention will become apparent from the following detailed description of the presently preferred embodiment, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a 4-node server system according to the invention.

FIG. 2 is flow diagram of a spin mutex used in the system of FIG. 1.

FIG. 3 is flow diagram of a sleep mutex.

FIG. 4 is a flow diagram that depicts panic handling in the system of FIG. 1.

FIGS. 5 and 6 are flow diagrams that depict mutex handling in the system of FIG. 1.

FIG. 7 shows a computer software program product embodying the invention.

FIGS. 8A and 8B show the improved mutex data structure of the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Technical Background

In general, all objects within an OS (or any other software) need a unique identifier (ID), which can be used to refer to the object in an efficient manner. For example, objects stored in an array may use the array index as the ID. Other objects like threads and processes, use a generated ID that is assigned to the object at the time the object is created (or initialized for activity). Processor (engine) objects are also assigned an ID. Since processors (physically) and thread/processes (logically) are actually doing the work of the computer system, these objects/entities can be said to "own" various other objects or entities in the system. When these relationships are being tracked, the objects IDs are used as "owner IDs".

The present invention includes in the mutex data structure an indicator of whether the mutex was acquired pre-panic or post-panic, preferably by modifying the engineID after the panic is initiated, preferably by assigning the engines different engineIDs post-panic.

The present invention also is a method of using this pre-panic or post-panic indicator during the panic shutdown. The present invention checks the pre-/post-panic indicator (for example, the engine ID) of a mutex, and overrides, waits for, or otherwise handles the mutex differently if it is pre-panic than if it is post-panic.

FIG. 1 shows a representation of a four node non-uniform memory access (NUMA) SMP system 10 with identical replicated OS instances 13. The present invention deals with the panic/shutdown handlers 11 and mutexing (mutual exclusion) primitives 12 of an SMP operating system (OS).

In this example, each processor is represented in the SMP OS 13 with one or more processor-private data structures. In addition, it is assumed that mutexes are implemented in one or more data structures. For the purpose of the present invention, these data structures are extended to include information about the state of the processor and mutex, specifically whether or not the system is in a panic/shutdown state.

Spin Mutex

FIG. 2 depicts a spin mutex: A spin mutex is a mutex whose wait mechanism involves simple looping without any throttling mechanism. A processor that is attempting to acquire a spin mutex that is held by another will consume the maximum amount of processor time spinning in a tight loop while repeatedly checking if the mutex is free.

Referring to FIG. 2, when attempting to acquire a spin mutex (p_mutex_spin) 21, a tight loop is entered whose only exit criteria is the mutex being free (available) 22. Once the exit criteria are met, the mutex is acquired (held) 23 and the identifier of the processor/thread/process (who acquired the mutex) is stored in the mutex data structure 24. When releasing a spin mutex (v_mutex_spin) 25, the mutex is simply set to a state of “free” (i.e. available) 26.

Sleep Mutex

FIG. 3 depicts a sleep mutex. An integral part of acquiring a sleep mutex is the sleeping, which generally implies a potential change of running processor. The pertinent information for the mutex is the processor ID of the processor that acquired the mutex, not the processor that might be running that thread at any given time. Therefore, the processor ID should be used as the mutex ID, set at the time that the mutex was acquired. The purpose of storing this information in the mutex is to provide a target for shoot-down in case this mutex is found to be held after the system enters the post-panic state. It is an implementation choice to either track the processor that acquired the mutex or the processor that last ran the thread holding the mutex. This choice drives whether to use the processor ID or the thread ID. Since the thread ID is always useful for debugging purposes, it may be practical to store both.

As shown in FIG. 8A, the processor ID is preferably an eight bit value corresponding to the number of the processor in the system. The processor ID is modified to create the owner ID 72 of the mutex data structure 70 by flipping the high order bit of the processor ID. Thus, processor number five would be assigned processor ID 00000101 (the binary value equal to the decimal number “five”), and the corresponding mutex owner ID 72 would be 10000101. The mutex data structure 70 also

contains additional parameters 74 specific to the type of mutex represented, but those parameters are not pertinent to the invention and are not described further.

When attempting to acquire a sleep mutex (p_mutex_sleep) 31, a loop is entered whose only exit criteria is the mutex being free (available) 32. However, unlike the spin mutex, after each test of the mutex for availability, the thread is put to sleep 33. The only way to be awaked from this sleep is for another thread to release the mutex. This means that the processor running the p_mutex_sleep is capable of running some other thread while this thread waits for the mutex. Once the exit criteria are met, the mutex is acquired 34 and the processor ID identifying the processor/thread/process (who acquired the mutex) is modified 35 to create the mutex ID 72 which identifies the owner of the mutex, and is stored 35 in the mutex data structure 70. When releasing a sleep mutex (v_mutex_sleep) 36, the mutex is set to a state of "free" (i.e. available) 37 and any threads sleeping for this mutex are awakened (variations include waking all or only one of the sleepers) 38.

Various hybrids of spin and sleep mutexes are possible, such as spinning for a period of time and then sleeping. Some variations use a time-delay instead of sleep, or in addition to it.

Note that both FIGS. 2 and 3 depict only general actions. Typically, much if not all of the above will have to be done atomically. In addition, there may be changes made to interrupt priority levels, etc. How this is accomplished is platform dependent. Additionally, there may be sanity checking such as self-deadlock, hang (timeout), and stolen mutex detection. All of these are well known and well within the skills of one having ordinary skill in these arts.

25

Panic Handling of a Shutdown

FIG. 4 shows the state transition of a general system shutdown implementing this invention, more specifically, a panic shutdown. Although both panic and "voluntary" shutdowns are handled similarly, with respect to mutexes, only the panic case is described for illustration purposes.

Once a panic is initiated 41, the panicking processor changes its flag to indicate that it is legitimately operating in the panic state 42, and changes the mutex routines used by the system to be the ones for panic 43. An additional task is the setting of some global system state or flag to indicate that the system is panicking.

5 Other standard panic activities might include shooting down all other processors 44, general clean up of the system 45 (possibly including flushing buffers and generating a core dump), and then returning to firmware control 46 which may initiate a reboot, or simply halt or power-off.

10

Mutex Handling, Normal vs. Panic

FIG. 5 depicts mutex handling implementing this invention. During normal system operation, acquiring a mutex 51 consists of the normal process for that mutex, plus the setting of a mutex flag 52 to indicate the mutex was acquired while the system was in the "normal" state.

During a panic 54, acquiring a mutex consists of checking if the running processor has been transitioned to the panic state ("does the processor object have its panic flag set?") 55 and if not, then jumping directly to the processor shutdown code 56, and checking the mutex for having been acquired in the "Normal" system state 58 in which case the processor that holds the mutex is shot-down 59 (perhaps redundantly) and the mutex is forced to be free 60. Once a mutex is acquired 61, the mutex flag is set to indicate that it was acquired during the "Panic" system state 62. This indicator of a mutex's post-panic acquisition is accomplished by modifying the identifier of the owner of the mutex.

As seen in FIG. 8B, a post-panic mutex's mutex ID 72' is different than the identifier 72 of a pre-panic mutex, in that the two high-order bits of the pre-panic mutex ID 72 are flipped. Thus, a mutex owned by processor number 5 would be given a pre-panic mutex ID 72 equal to 10000101 as described above and shown in FIG. 8A, but would be give a post-panic mutex ID 72' equal to 01000101 as shown in FIG. 8B. The other parameters 74 of the post-panic mutex data structure 70' are not changed from their pre-panic values. Of course, one of ordinary skill will understand that various other ways

25

30

exist to include a pre-/post-panic indicator in a mutex data structure, or in an external data structure referenced by the mutex handling routines.

Referring now to FIG. 6 during normal system operation 63, releasing a mutex 64 consists of the normal process for that mutex. During a panic 65, releasing a mutex 5 consists of checking the running processor has been transitioned to the panic state (does the processor object have its panic flag set?) and if not then jumping directly to the processor shutdown code 67, and the normal process 64 for that mutex. Note however, that while the mutex ID and flag could be cleared, their contents are not valid unless the mutex is held, and so clearing the mutex ID and flag would be unnecessary.

10

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
18

Advantages Over the Prior Art

This invention permits determination of the context that a mutex was acquired under (pre-panic vs. post-panic) which is not available in other mutex mechanisms.

5 Other mechanisms that are used to resolve shutdown mutex problems, have been limited to timeouts, which have inherent problems including having to guess at how long is "too long" to wait for a mutex and how to measure such a time period in an efficient manner at such a low level, in addition to not providing any intelligence about whether the mutex was legitimately acquire post-panic.

10

Alternative Embodiments

It will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the invention. In particular, if a state transition invalidates the threads executing prior to the transition, then the mutexes must also be transitioned (invalidated). If the state transition is such that all threads are notified and voluntarily terminate, then there is no need for mutex handling. However, if the transition is forced (e.g. from normal operation to a panic; from single uses to multi-user mode; between run levels) then it is necessary to properly handle not only the mutexes but also any threads that failed to make the transition. The present invention can therefore be extended to include state transitions of not only processors and systems, but also threads and processes. Accordingly, the scope of protection of this invention is limited only by the following claims and their equivalents.